0: Добавить свою папку в GIT.

1: Переписать лабораторную работу «Peer-2-peer чат» с использованием WCF. Каждый запущенный экземпляр программы может подключаться к произвольному числу других пользователей в произвольный момент времени, и аналогично другие пользователи могут подключаться к нему. Сообщение, созданное одним из участников образовавшейся сети, должно быть передано всем текущим её участникам. Предусмотреть возможность существования циклов в топологии сети при передаче собщений.

2: Написать программу на WPF или WinForms, которая позволяет загружать изображение, показывать его на форме и применить к нему произвольный фильтр с отображением результата. При этом степень готовности операции отображается на контроле вроде ProgressBar. Во время использования фильтра интерфейс должен быть отзывчивым, то есть, например, можно нажимать на кнопки.

3: Файберы – это потоки внутри потока, которые надо переключать вручную (<https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms682661%28v=vs.85%29.aspx>). Пример их реализации доступен в проекте Fibers. ProcessManagerFramework.cs содержит модель процесса операционной системы, где периоды работы сменяются периодами операций ввода-вывода, и при этом априори невозможно узнать, какова будет длительность этих интервалов.

В операционных системах Windows 3.1 и ранних MacOS не было понятия вытесняющей многозадачности. Вместо этого управление отдавалось другому процессу добровольно и вручную – например, когда процесс находился в ожидании операции ввода-вывода. Это реализовано в модели процесса.

Требуется:

* Создать несколько экземпляров класса Process;
* Сделать две реализации метода ProcessManager.Switch – с приоритетным и бесприоритетным алгоритмом диспетчеризации;
* По завершению всех процессов удалить все файберы кроме основного и корректно выйти из программы.

4: Для указанного алгоритма:

* Определить сложность его последовательной реализации;
* Предложить схему распараллеливания;
* Описать её в виде графа «операции-операнды»;
* Оценить ускорение и эффективность, а также определить число процессоров, начиная с которого будет достигнуто максимальное ускорение. Вычислить это ускорение и эффективность.

Алгоритмы:

* Скалярное произведение двух N-мерных векторов;
* Среднее арифметическое N чисел;
* Минимум из N чисел;
* Евклидова мера N-мерного вектора:

5: Реализовать решение упрощённой задачи «производитель-потребитель» с указанным средством синхронизации:

* Производители – объекты, кладущие некоторые объекты (например, числа, строки или более сложные объекты-заявки) в экземпляр класса List<T>;
* Потребители – объекты, извлекающие заявки из экземпляра List<T>;
* Между двумя добавлениями и двумя изъятиями вставляется пауза;
* Количество производителей и потребителей задаётся константами;
* При запуске программы создаются производители и потребители. Они прекращаюют работу по нажатию произвольной клавиши. При этом завершение работы производителей и потребителей должно быть корректно реализовано (Thread.Kill таковым не является).

6: Реализовать объект ThreadPool, реализующий паттерн «пул потоков». Число потоков задаётся константой в классе. Добавление задачи осуществляется с помощью нестатического метода Enqueue(Action a). Класс должен быть унаследован от интерфейса IDisposable и корректно освобождать ресурсы при вызове метода Dispose().

7: Реализовать веб-сервис с использованием технологии WCF или альтернативной, позволяющий решать одну из следующих или подобных им задач:

* Сортировка массива чисел;
* Нахождение всех простых чисел меньше заданного;
* Умножение матриц;

Провести нагрузочное тестирование данного сервиса. Допускается как размещение сервиса на одном хочте, а клиентов на другом, так и тестирование в рамках одного хоста.

8: Реализовать алгоритм из задачи 7 с применением MPI.NET. Распараллеливанию подлежит реализация алгоритма, то есть все узлы MPI работают над частями одного запроса. Предусмотреть корректное завершение работы отдельных процессов.

9: Реализовать алгоритм, считающий факториал числа в 4 потоках. Алгоритм должен быть реализован с использованием неблокирующей синхронизации по типу “без ожидания”.

**Работы с закоммиченными папками \bin, \obj и файлами с расширением .suo не принимаются.**